



UNIVERSIDAD DE GRANADA

TRABAJO FIN DE GRADO

INGENIERÍA INFORMÁTICA

Aprendizaje Automático para la extracción de características y detección de situaciones anómalas en multitudes

Autor

Diego Navarro Cabrera

Directores

Name of the main supervisor

Name of the second supervisor (if available)



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA Y
TELECOMUNICACIONES

—
Granada, 3 de mayo de 2021

Aprendizaje Automático para la extracción de características y detección de situaciones anómalas en multitudes

Diego Navarro Cabrera

Palabras clave:

Resumen

Machine learning for feature extraction and abnormal crowd behavior

Diego Navarro Cabrera

Keywords:

Abstract

Yo, **Diego Navarro Cabrera**, alumno del Grado de Ingeniería Informática de la **Escuela Técnica Superior de Ingeniería Informática y Telecomunicaciones de la Universidad de Granada**, con DNI 75935043Z, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

A handwritten signature in blue ink, appearing to read "Diego", enclosed within a roughly oval-shaped outline.

Fdo: Diego Navarro Cabrera

Granada, 3 de mayo de 2021

D. Name of the main supervisor y D.^a Name of the second supervisor (if available), profesores del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado *Aprendizaje Automático para la extracción de características y detección de situaciones anómalas en multitudes*, ha sido realizado bajo su supervisión por **Diego Navarro Cabrera**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 3 de mayo de 2021

Los directores:



Name of the main supervisor



**Name of the second supervisor
(if available)**

Agradecimientos

Índice general

Índice general	5
1 Introducción	7
2 Fundamentación Teórica	8
2.1. Aprendizaje automático	8
2.2. Máquina de soporte vectorial	8
2.3. Extracción de características	8
2.4. Autoencoders	8
2.5. Aprendizaje multi-tarea	8
3 Descripción del problema	9
3.1. Conjuntos de datos de trabajo	9
3.1.1. UMN Crowd Dataset	10
3.1.2. Violent Flows	11
3.2. Estado del arte	12
3.2.1. Modelo basado en descriptores visuales locales de nivel medio	13
3.3. Modelo propuesto	14
3.3.1. Representación de la multitud	15
3.3.2. Descriptores visuales	17
3.3.3. Vector de características y clasificador	17
4 Resultados experimentales	18
4.1. Metodología usada	18
4.1.1. UMN Crowd Dataset	18
4.1.2. Violent Flows	18
4.2. Comparación de resultados	18
4.2.1. UMN Crowd Dataset	18
4.2.2. Violent Flows	18

5 Conclusión	19
Bibliografía	20

1 Introducción

Intro

Fundamentación Teórica

- 2.1. Aprendizaje automático
- 2.2. Máquina de soporte vectorial
- 2.3. Extracción de características
- 2.4. Autoencoders
- 2.5. Aprendizaje multi-tarea

3

Descripción del problema

En este capítulo desarrollaremos en más profundidad el problema a tratar. Como ya hemos mencionado anteriormente, nuestro objetivo es estudiar el comportamiento de una multitud para detectar cuándo se está produciendo una anomalía.

En concreto nos centraremos en el enfoque holístico o *top-down* de este problema. Este enfoque trata a la multitud como un solo ente y analiza la dinámica del grupo más que las acciones de sus individuos. La detección de comportamientos individuales anómalos dentro de la multitud queda fuera del ámbito de este trabajo, pero algunas propuestas de las estudiadas podrían adaptarse para ello.

La estructura de esta sección será la siguiente. Primero hablaremos de los conjuntos de datos sobre los que trabajaremos. A continuación trataremos el estado del arte y explicaremos en más profundidad el modelo sobre el que nos hemos basado. Finalmente detallaremos el modelo desarrollado y diversas alternativas que se han tenido en cuenta.

3.1. Conjuntos de datos de trabajo

Se han elegido 2 conjuntos de datos ampliamente usados en el ámbito de la detección de anomalías en multitudes. El primero es el UMN Crowd Dataset [1], que consiste de una pequeña selección de vídeos en el que un grupo de personas echa a correr tras una señal. El segundo dataset es conocido como Violent Flows [2] y recoge una mezcla de vídeos de peleas y vídeos de control. Estos dos conjuntos de datos nos permitirán comprobar la efectividad del modelo ante 2 de las anomalías más habituales y relevantes en el día a día: las peleas y escenarios de pánico. Además su amplio uso en la literatura nos permitirá comparar fielmente nuestro modelo con otros muchos . A continuación describiremos más detalladamente ambos conjuntos.

3.1.1. UMN Crowd Dataset

Este conjunto de datos, elaborado por la Universidad de Minnesota (UMN), consiste de 11 vídeos en los que un grupo de personas se comporta de manera normal hasta que recibe una señal y todos se dispersan corriendo. Los 11 vídeos se dividen en 3 escenas distintas, cada una con 2, 6 y 3 vídeos respectivamente. Todos estos vídeos se encuentran unidos en un mismo archivo que conviene dividir antes de ser procesado.



Figura 3.1: Ejemplo de fotograma normal (izqda) y anómalo (drcha) en la escena 1.

Este dataset clasifica los fotogramas normales y los anómalos por medio de un cartel que aparece en pantalla cuando se produce la anomalía, tal y como se puede ver en 3.1. Para evitar que esto pueda influir en nuestro modelo se ha recortado la parte superior de cada vídeo para eliminar dicho cartel. Por otro lado, tal y como se detalla en [3] estos carteles no son exactos, ya que aparecen significativamente después del comienzo de la anomalía. Es por esto que se ha decidido usar las anotaciones provistas en dicho artículo para etiquetar el comienzo de la anomalía en cada vídeo. Además de esto, en la mayoría de los casos la anomalía acaba antes de que empiece el siguiente vídeo, por lo que también tendremos que anotar el fotograma de finalización de cada caso. Hablaremos con más detalle de esto en el apartado de experimentación.

A pesar de sus ventajas y su uso muy extendido, este dataset presenta algunos inconvenientes que hay que tener en cuenta. Para empezar, su tamaño de 11 vídeos es bastante limitado, más aún si tenemos en cuenta que están divididos en 3 escenas que han de evaluarse de manera separada. Además solo representa anomalías de un tipo muy específico (gente echando a correr), por lo que no nos sirve para evaluar modelos enfocados a detectar todo tipo de

anomalías, algo de lo que hablaremos cuando propongamos nuestro modelo.

3.1.2. Violent Flows

Este conjunto de datos es una colección de vídeos de escenas reales para entrenar y evaluar modelos de detección de violencia. Contiene 246 instancias que se dividen en vídeos con violencia y sin ella. Se tratan de escenas cortas de entre 1 y 6 segundos y que se clasifican de manera íntegra como violentas o no.



Figura 3.2: Fotograma de ejemplo de una escena normal (izqda) y una violenta (drcha)

Los vídeos de este conjunto han sido extraídos de YouTube, y son muy distintos entre si, lo que nos permitirá comprobar la capacidad de generalización de nuestro programa. Por otro lado, la baja resolución a la que se encuentran dificulta su análisis, ya que algunos fotogramas están tan borrosos que es difícil saber qué está pasando.

De forma similar al conjunto de datos anterior, este posee una cantidad de instancias de entrenamiento bastante limitada (246 vídeos clasificados individualmente), y se centra en un tipo de anomalías concreto, en este caso las peleas. Por otro lado, al contrario de lo que cabría esperar al trabajar con anomalías, ambas clases tienen un número de instancias muy equilibrado, 121 vídeos violentos y 125 normales, lo que nos podría facilitar el entrenamiento de un modelo de clasificación binaria, pero dificultar el uso de otras opciones centradas en entrenar únicamente con instancias normales (clasificadores de una sola clase).

3.2. Estado del arte

Existen numerosos trabajos que han probado diversas formas de detectar comportamientos anómalos. Algunas de estas son:

- Estudiar el flujo óptico de la imagen. Por ejemplo, en [4] usan el valor medio de este para estimar una interacción de fuerzas y detectar posibles zonas de conflicto, tal y como se ve en la figura 3.3.



Figura 3.3: Demostración de la interacción de fuerzas, primero creamos un grid de partículas y estudiámos su trayectoria interpolando los movimientos cercanos a estas. Las zonas con mayor movimiento en una misma dirección se marcan como más fuertes, y en función de dichas fuerzas se detectan comportamientos anómalos.

- Codificar el vídeo por medio de un modelo entrenado con vídeos no anómalos. Puesto que dicho modelo solo conoce los vídeos normales, al intentar reconstruir una anomalía se obtendrá un resultado más irregular o con una mayor tasa de error, lo que nos marcará que se está produciendo una anomalía. Un ejemplo de esto se puede ver en [5] que usa un diccionario de códigos binarios y un histograma de dichos códigos como representación.
- Usar modelos end-to-end de deep learning como en [6] que entrena 2 redes adversarias: un generador y un discriminador, y usa el discriminador para clasificar cada frame como normal o anómalo.

En nuestro caso queremos centrarnos en un modelo que extraiga un conjunto de características de forma analítica, aprovechando el conocimiento y la intuición que ya tenemos sobre el tipo de anomalías esperadas. Dentro de los trabajos que siguen este enfoque el que parece aportar más información y mejores resultados es el que trataremos a continuación.

3.2.1. Modelo basado en descriptores visuales locales de nivel medio

El modelo sobre el que vamos a basar nuestra propuesta es el expuesto en [7]. Este artículo propone un proceso analítico de extracción de características, que se realiza a partir de una representación espacio-temporal de la multitud. Esta representación está formada por un conjunto de puntos de interés extraídos por un algoritmo de detección como FAST o SIFT.

Los puntos son rastreados a lo largo del tiempo, guardando las trayectorias que siguen, representando así la información temporal del modelo. Para hacer esto se usa un rastreador de características dispersas como ROLF o el rastreador de Kanade–Lucas–Tomasi.

Para representar la información espacial, cada punto está relacionado con sus vecinos por medio de una triangulación de Delaunay. Dicha triangulación es una red de triángulos conexa y convexa en donde la circunsferencia circunscrita de cada triángulo no contiene ningún vértice de otro triángulo. También podría usarse un algoritmo de los K vecinos más cercanos, pero el artículo sostiene que el grafo aporta una mejor representación de la información local y espacial.

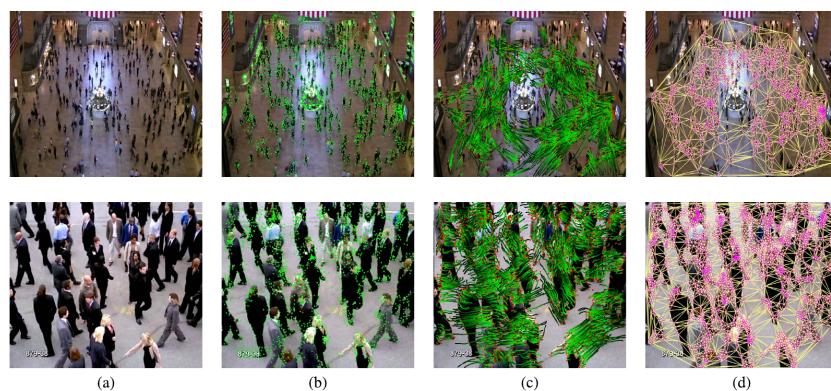


Figura 3.4: Ejemplo extraido de [7]. (a) Fotograma de ejemplo. (b) Detección de puntos de interés con FAST. (c) Trayectorias de los puntos a lo largo del tiempo. (d) Triangulación de Delaunay.

Una vez representada la información de la multitud, se calcula una serie de descriptores por cada punto detectado. Los descriptores son los siguientes:

- **Velocidad** de cada punto rastreado.

- **Variación del flujo dirección:** cuanto más recta sea la trayectoria, menor será este valor.
- **Estabilidad** de la estructura de la multitud: una multitud que mantenga una misma forma durante mucho tiempo será estable, pero una que se mueve de forma errática no.
- **Cohesión** (*Collectiveness*): grado en el que los individuos se mueven de manera conjunta y en una misma dirección.
- **Conflictos:** medición del nivel de interacción entre personas cercanas.
- **Densidad local:** número aproximado de personas en una determinada zona, calculado en función del número de puntos localizados en esa zona.
- **Uniformidad:** indicador de si los puntos rastreados se distribuyen de manera uniforme o se concentran en ciertas zonas formando subgrupos.

Una vez hechos todos los cálculos se generan histogramas para cada descriptor, se juntan para formar un vector de características y se entrena un SVM de clasificación binaria.

En el propio artículo podemos ver los resultados de este modelo en los 2 conjuntos de datos que vamos a usar, obteniendo una puntuación de 88%, usando la métrica del área por debajo de la curva (AUC), en el conjunto de Violent Flows, y un 98.6% en el de UMN. Estos resultados son especialmente buenos, sobre todo si nos fijamos en las comparaciones que se exponen en el mismo artículo con respecto a trabajos similares.

Por otro lado, también menciona la capacidad de este sistema de funcionar a tiempo real, aunque solo indica que siendo implementado en C++ funciona a 5 fotogramas por segundo “*sin estar optimizado*”. Esta afirmación es demasiado vaga como para ser tomada en cuenta, ya que la velocidad de este algoritmo depende en gran medida del número de puntos captados, algo que puede variar en gran medida en función del vídeo o de los parámetros elegidos. Más adelante expandiremos más sobre este tema.

3.3. Modelo propuesto

En esta sección hablaremos del proceso de implementar un sistema lo más cercano posible al descrito en el apartado anterior, así como los cambios

que se han probado para mejorarlo. La mayoría de lo que se mencione en este apartado será común tanto al modelo descrito en [7], como a nuestro modelo, ya que estamos partiendo de este, pero se señalarán los casos en los que se diverja del diseño original.

El mayor obstáculo que se ha encontrado para intentar replicar este modelo es que aunque está lo suficientemente bien explicado como para entender su funcionamiento, a la hora de entrar en detalles de implementación, e incluso de evaluación sobre los conjuntos de datos, hay apartados en los que no disponemos de suficiente información como para replicar fielmente lo expuesto en el artículo. iremos mencionando estos casos conforme vayamos avanzando.

3.3.1. Representación de la multitud

Como comentábamos en la sección anterior, el primer paso para detectar una anomalía es representar la información espacio-temporal de la multitud. Para esto primero detectamos un conjunto de puntos de interés mediante el algoritmo FAST (Features from Accelerated Segment Test) [8]. El fin de este algoritmo es buscar esquinas en la imagen, y dada la forma y apariencia de una persona, se detectará un gran número de estos puntos en cada persona. Para distinguir a los puntos que corresponden con una persona de los del fondo de la imagen usaremos un filtro basado en su velocidad de movimiento. Puesto que la velocidad de cada punto es uno de los descriptores que vamos a usar, podemos usar este valor para eliminar todos los puntos que se encuentren por debajo de un umbral. Este filtro suele funcionar bien en escenas en las que la cámara está fija, pero como veremos más adelante este no siempre es el caso, e incluso hay factores que pueden complicar su buen funcionamiento aunque se cumpla esta condición.

Una vez tenemos un conjunto de puntos localizados usaremos el rastreador de Kanade–Lucas–Tomasi, que nos puede indicar la posición siguiente de un punto a partir de dos fotogramas consecutivos y su localización original. Para una mayor fiabilidad en el rastreo usaremos un esquema de verificación hacia delante y hacia atrás, es decir, después de obtener la posición final de un punto la verificaremos realizando el rastreo a la inversa, si obtenemos de nuevo la posición original tomaremos el resultado como bueno, y si no consideramos que hemos perdido la trayectoria y eliminaremos el punto.

La forma de gestionar las trayectorias será el primer aspecto en el que

nos separaremos de [7]. En este artículo se menciona una alternancia entre la detección de nuevos puntos y el rastreo de los ya existentes en función de la fiabilidad de la trayectoria, sin embargo demasiados aspectos se dejan en el aire, como la manera de incorporar estos nuevos puntos en el sistema sin sobrecargarlo de posiciones muy cercanas entre si, o el criterio para detectar a personas que hayan podido entrar en escena.

Es por eso que en su lugar se ha decidido seguir un esquema más sencillo en el que si usamos trayectorias de longitud L , primero llenamos la trayectoria del punto durante esos L fotogramas y luego calculamos los descriptores de ese punto durante otros L fotogramas. De esta forma, detectariamos nuevos puntos, como mucho, cada $L * 2$ fotogramas. Puesto que este método nos obligaría a renunciar a la información de la mitad de los fotogramas usaremos 2 conjuntos de trayectorias, una que usaremos para calcular los descriptores y otra en la que iremos construyendo la trayectoria de los nuevos puntos.

Cabe destacar que este segundo conjunto de trayectorias podría deshecharse en una implementación orientada a un uso práctico en la que no fuese importante obtener toda la información de los vídeos, pero los conjuntos de datos con los que trabajamos son bastante pequeños y las precisiones lo bastante altas como para que cada fotograma cuente.

Por otro lado, como decíamos en el apartado anterior, usaremos una triangulación de Delaunay para relacionar espacialmente los puntos de cada fotograma. OpenCV ya nos proporciona una clase que llevará a cabo este proceso, llamada SubDiv2D. Una vez creado el grafo en los siguientes apartados haremos uso del concepto de clique de primer orden para relacionar unos puntos con otros. Definimos el clique del punto P_i como todos los puntos P_j del grafo tales que existe una arista que une P_i y P_j . En [7] se habla también de los cliques de orden mayor que uno. Estos cliques se definen de manera recursiva de forma que un clique de orden k se forma añadiendo al clique de orden $k-1$ los puntos conectados con cualquier miembro este. Puesto que en ningún momento se menciona que estos aporten nada de valor al cálculo, y además añadirían un gran peso computacional, ignoraremos estos cliques de orden superior y usaremos solo cliques de primer orden.

Resumiendo todo el proceso de este apartado nos quedamos con 3 pasos principales. Primero captar puntos de interés con FAST, que más adelante serán filtrados en función de su velocidad. Segundo, representar la evolución

temporal de la multitud por medio de las trayectorias de cada punto a lo largo de un determinado número de fotogramas. Tercero, representar las relaciones espaciales de los puntos en cada fotograma por medio de los cliques de primer orden presentes en una triangulación de Delaunay que contenga a todos los puntos.

3.3.2. Descriptores visuales

En este apartado explicaremos en mayor profundidad los descriptores que usaremos para representar el estado de la multitud. Estos se calculan usando a partir de las representaciones explicadas en el apartado anterior. Existen 3 parámetros que determinarán que momento de la trayectoria usamos para realizar ciertos cálculos. El primero es el tamaño total de la trayectoria, que denotaremos como L . Puesto que estas trayectorias miden un periodo de tiempo relativamente amplio también usaremos otros dos parámetros, τ_1 y τ_2 , que marcarán momentos intermedios de la trayectoria que usaremos como referencia a la hora de calcular algunos valores como la velocidad.

- **Velocidad:** en [7] se calcula este valor como la distancia euclídea de la posición actual de un punto y su posición τ_1 fotogramas antes. También se menciona un mapa de perspectiva creado interpolando la altura de una persona de referencia en dos extremos de la imagen. Esta idea está insuficientemente explicada como para poder ser replicada con fidelidad, y además solo es aplicable a conjuntos de datos en los que los videos comparten una misma escena. Es por esto que en su lugar se ha decidido dividir este descriptor en 2 que se traten por separado, uno que mida la velocidad en el eje x y otro que la mida en el eje y . Esta división permitiría interpretar mejor escenas en las que la cámara no está perpendicular al plano en el que se mueve la multitud, como ocurre en la mayoría de los casos. En cuanto al filtro que mencionábamos en el apartado anterior, una vez calculada la distancia que recorre un punto en el eje x e y , si la suma de ambos valores (distancia Manhattan) es menor que un valor concreto β , dicho punto es eliminado.

3.3.3. Vector de características y clasificador

Resultados experimentales

4

4.1. Metodología usada

4.1.1. UMN Crowd Dataset

4.1.2. Violent Flows

4.2. Comparación de resultados

4.2.1. UMN Crowd Dataset

4.2.2. Violent Flows

5 Conclusión

Conclusión

Bibliografía

- [1] UMN Crowd Dataset. http://mha.cs.umn.edu/proj_events.shtml#crowd/.
- [2] Violent Flows Dataset. <https://www.openu.ac.il/home/hassner/data/violentflows/>.
- [3] Duan-Yu Chen y Po-Chung Huang. "Motion-based unusual event detection in human crowds". En: *J. Visual Communication and Image Representation* 22 (feb. de 2011), págs. 178-186. doi: [10.1016/j.jvcir.2010.12.004](https://doi.org/10.1016/j.jvcir.2010.12.004).
- [4] R. Mehran, A. Oyama y M. Shah. "Abnormal crowd behavior detection using social force model". En: (2009), págs. 935-942.
- [5] Mahdyar Ravanbakhsh y col. "Plug-and-Play CNN for Crowd Motion Analysis: An Application in Abnormal Event Detection". En: (oct. de 2016).
- [6] Mahdyar Ravanbakhsh y col. "Training Adversarial Discriminators for Cross-channel Abnormal Event Detection in Crowds". En: (jun. de 2017).
- [7] Hajar Fradi, Bertrand Luvison y Quoc-Cuong Pham. "Crowd Behavior Analysis Using Local Mid-Level Visual Descriptors". En: *IEEE Transactions on Circuits and Systems for Video Technology* PP (oct. de 2016), págs. 1-1. doi: [10.1109/TCSVT.2016.2615443](https://doi.org/10.1109/TCSVT.2016.2615443).
- [8] Edward Rosten, Reid Porter y Tom Drummond. "Faster and Better: A Machine Learning Approach to Corner Detection". En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.1 (2010), págs. 105-119. doi: [10.1109/TPAMI.2008.275](https://doi.org/10.1109/TPAMI.2008.275).